

Research on 2-D SDCT Algorithm Based on Parallel Computing

Wenbang Sun^{1,2}, Hexin Chen¹, Hong Chen², Zhiqiang Wang²

¹School of Communication Engineering, Jilin University Changchun China

²Aviation Information Department, Aviation University of Air Force Changchun China
chwenbang@163.com; chx@jlu.edu.cn; chhong1023@163.com; kaola0811@126.com

Abstract—At present, 2-D DCT is applied widely in the field of signal processing. But the transform actually operates 1-D DCT to the rows and columns of 2-D data successively, which limits the transform speed to farther improve. To overcome such drawback, a parallel computing method is proposed in the paper. First, some new matrix operation algorithms and a new transform matrix are defined. Then, 2-D SDCT (Submatrix Discrete Cosine Transform) is operated integrally based on the new transform matrix and new matrix operation algorithm. Finally, the parallel computing of 2-D DCT is analyzed based on the characteristic of 2-D SDCT. The theoretical analysis shows that the calculating amount of the proposed method only needs one time multiplication and a few times additions, and the transform speed relative to other fast algorithms is improved notably.

Keywords—2-D SDCT, Fast Algorithm, Parallel Computing, Serial Computing.

I. INTRODUCTION

According to mean square error (MSE) criteria, K-L is the optimal orthogonal transform for first-order Markov process at present^[1]. But the transform needs statistical information of the sampled data and hasn't fast algorithm. However, DCT doesn't depend on signal itself, transform performance is very close to the K-L and suitable for hardware processing. So the DCT is applied widely in the field of digital signal processing, and especially DCT has become the core part of motion or static image compression standard, such as JPEG, MPEG, H.26x etc.

But the computing process is complex, which needs a large amount of multiplication operation and addition operation, affects the transform efficiency markedly. The calculating amount will be tremendous and is hard to reach real time processing if the DCT is applied directly without improvement.

According to the symmetry of transform matrix, Chen^[2] firstly proposed a fast DCT algorithm based on sparse matrix decomposition in 1977. And after that, many better fast algorithms has occurred successively^[3-10]. The most influential algorithms are Vetterli algorithm [8], Feig algorithm [9] etc. Most of the fast 2-D DCT algorithms perform transform by twice 1-D DCT successively, and aim to reduce times of addition operation and multiplication operation, especially multiply operation. Although the times of addition operation and multiply operation can be reduced to a certain extent, twice 1-D DCT operated successively will still limit to improve further the transform speed.

At the present time, software technology and hardware technology related to parallel computing are increasingly mature, and the parallel computing is becoming the main theme of scientific and engineering computation in the 21st century, so

the research on fast 2-D DCT based on parallel computing appears more significant. Therefore, a parallel computing idea of 2-D SDCT (Submatrix Discrete Cosine Transform) is proposed in the paper, which input the 2-D signal integrally and output the transform result integrally in order to increase the computing speed.

The paper is organized as follows: The next section describes 2-D matrix expression and matrix operation method. Section III describes traditional operation method of 2-D DCT. Section IV describes 2-D SDCT algorithm. Section V discusses the parallel computing of 2-D SDCT in detail. Section VI analyses the algorithm performance of 2-D SDCT. The last section gives some conclusions.

II. MATRIX EXPRESSION AND OPERATION

In order to describe conveniently, matrix expression form and some matrix operation methods need to define first^[11, 12].

A. Matrix Expression

In order to express conveniently, the bold letters, **A**, **B** etc or the abbreviation, $[a_{ij}]$, $[b_{ij}]$ etc, are adopted to describe a matrix, where i and j represent the row and column of an element in 2-D matrix. When total number of rows, R , and total number of columns, C , are necessary to state, $\mathbf{A}_{R \times C}$ or $[a_{ij}]_{R \times C}$ is adopted to denote. In addition, a larger matrix can be divided into sub-matrices (or sub-blocks). The matrix with sub-matrix for element is called partitioned matrix.

The sub-matrix can be expressed as $\mathbf{A}_{rc, R \times C}$, where, $R \times C$ indicates sub-matrix size, the r and c indicate sub-matrix position in partitioned matrix. A certain element in sub-matrix be expressed as $a_{ij, rc}$, where, r and c indicate sub-matrix position in partitioned matrix, i and j indicate an element position in sub-matrix. For example, if a 4×4 matrix is divided into 4 sub-matrices, equation (1) is adopted to describe.

$$\mathbf{A}_{4 \times 4} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11, 2 \times 2} & \mathbf{A}_{12, 2 \times 2} \\ \mathbf{A}_{21, 2 \times 2} & \mathbf{A}_{22, 2 \times 2} \end{bmatrix} \quad (1)$$

B. Matrix Operation Method

Definition 1: For a matrix, $\mathbf{A} = (a_{kl, rc})$, where $1 \leq r \leq R$, $1 \leq c \leq C$, $1 \leq k \leq K$ and $1 \leq l \leq L$, the "sub-matrix element position" operation of the matrix is marked as $\mathbf{A}^L$.

$$\mathbf{A}^L = (a_{rc, kl}) \quad (2)$$

The matrix $\mathbf{A}^L$ is called "position matrix" of matrix $\mathbf{A}$. For

example, for a 4×4 matrix, A , the matrix can be divided into 4 same type sub-matrices. The “sub-matrix element position” operation of the matrix can be described as follows:

$$(A_{4 \times 4})^L = \begin{bmatrix} a_{111} & a_{121} & a_{112} & a_{122} \\ a_{211} & a_{221} & a_{212} & a_{222} \\ a_{121} & a_{122} & a_{122} & a_{122} \\ a_{212} & a_{222} & a_{212} & a_{222} \end{bmatrix}^L = \begin{bmatrix} a_{111} & a_{112} & a_{121} & a_{122} \\ a_{121} & a_{122} & a_{122} & a_{122} \\ a_{211} & a_{212} & a_{221} & a_{222} \\ a_{212} & a_{212} & a_{222} & a_{222} \end{bmatrix}$$

Obviously, the operation is based on sub-matrix, and demands each sub-matrix be the same size.

Definition 2: For two matrices, $A=(a_{rc})$ and $B=(b_{rc})$, where $1 \leq r \leq R$ and $1 \leq c \leq C$, the “matrix element product” operation of the two matrices is marked as $A \nabla B$, and the “matrix element product” operation is defined as equation (3). The operation result is a 1×1 matrix, that is, the result is a constant

$$A \nabla B = \sum_{r=1}^R \sum_{c=1}^C a_{rc} b_{rc} \quad (3)$$

Definition 3: For two matrices, $A=(a_{rc})$ and $B=(b_{kl})$, where $1 \leq r \leq R$, $1 \leq c \leq C$, $1 \leq k \leq K$ and $1 \leq l \leq L$, the “sub-matrix product” operation of the two matrices is marked as $A \triangleright B$, and which result is a $K \times L$ matrix. The “sub-matrix product” operation is defined as equation (4).

$$A_{R \times C} \triangleright B_{(KR) \times (LC)} = A_{R \times C} \triangleright \begin{bmatrix} B_{11R \times C} & \cdots & B_{1LR \times C} \\ \vdots & \ddots & \vdots \\ B_{K1R \times C} & \cdots & B_{KL R \times C} \end{bmatrix} \quad (4)$$

$$= \begin{bmatrix} A_{R \times C} \triangleright B_{11R \times C} & \cdots & A_{R \times C} \triangleright B_{1LR \times C} \\ \vdots & \ddots & \vdots \\ A_{R \times C} \triangleright B_{K1R \times C} & \cdots & A_{R \times C} \triangleright B_{KL R \times C} \end{bmatrix}_{K \times L}$$

The “sub-matrix product” operation satisfies the following operation rules (Suppose A and B both be $R \times C$ matrix, and C be a $(KR) \times (LC)$ matrix):

$$(A+B) \triangleright C = A \triangleright C + B \triangleright C$$

Note: (1) In the processing, matrix B (larger matrix) needs to be divided into partitioned matrix according to the matrix A (small matrix) size, that is, the matrix A and the sub-matrices in matrix B are the same size.

(2) When the R , C and T are equal to 1, the “sub-matrix product” operation simplify as “matrix element product” operation. So the “matrix element product” operation is a special case of the “sub-matrix product” operation.

Definition 4: For two matrices, $A=(a_{rc})$ and $B=(b_{klrc})$, where $1 \leq r \leq R$, $1 \leq c \leq C$, $1 \leq k \leq K$ and $1 \leq l \leq L$, the “matrix superimposition product” operation of the two matrices is marked as $A \triangleleft B$, and which is defined as equation (5). The operation result is a $K \times L$ matrix.

$$A_{R \times C} \triangleleft B_{(KR) \times (LC)} = \begin{bmatrix} a_{11} & \cdots & a_{1C} \\ \vdots & \ddots & \vdots \\ a_{R1} & \cdots & a_{RC} \end{bmatrix} \triangleleft \begin{bmatrix} B_{11} & \cdots & B_{1C} \\ \vdots & \ddots & \vdots \\ B_{R1} & \cdots & B_{RC} \end{bmatrix} \quad (5)$$

$$= \left[\sum_{r=1}^R \sum_{c=1}^C a_{rc} B_{rc, K \times L} \right]_{K \times L}$$

The “matrix superimposition product” operation satisfies the following operation rules (Suppose A and B both be $R \times C$ matrix, and C be $(KR) \times (LC)$ matrix):

$$(A+B) \triangleleft C = A \triangleleft C + B \triangleleft C$$

Note: (1) In the processing of “matrix superimposition product” operation, matrix B (larger matrix) needs to be divided into partitioned matrix according to the element total of matrix A (small matrix), that is, the sub-matrix total of matrix B equal to the element total of matrix A .

(2) When the I , J and K all are equal to 1, the “matrix superimposition product” operation simplify as “matrix element product” operation. So the “matrix element product” operation is a special case of the “matrix superimposition product” operation too.

III. 2-D DCT ALGORITHM

Set the 2-D signal as $f(r, c)$, $r=0, 1, \dots, R-1$, $c=0, 1, \dots, C-1$, and the 2-D DCT are described by equation (6) and equation (7) respectively.

$$F(u, v) = \sum_{r=0}^{R-1} \sum_{c=0}^{C-1} f(r, c) \left\{ \sqrt{\frac{2}{R}} c(u) \cos \left[\frac{\pi(2r+1)u}{2R} \right] \right\} \quad (6)$$

$$\left\{ \sqrt{\frac{2}{C}} c(v) \cos \left[\frac{\pi(2c+1)v}{2C} \right] \right\}$$

$$f(r, c) = \sum_{u=0}^{R-1} \sum_{v=0}^{C-1} F(u, v) \left\{ \sqrt{\frac{2}{R}} c(u) \cos \left[\frac{\pi(2r+1)u}{2R} \right] \right\} \quad (7)$$

$$\left\{ \sqrt{\frac{2}{C}} c(v) \cos \left[\frac{\pi(2c+1)v}{2C} \right] \right\}$$

Where,

$$c(u) = \begin{cases} 1/\sqrt{2} & u = 0 \\ 1 & \text{other} \end{cases} \quad c(v) = \begin{cases} 1/\sqrt{2} & v = 0 \\ 1 & \text{other} \end{cases}$$

Because the 2-D DCT has separable transform properties, the operation of 2-D DCT can be accomplished by computing 1-D DCT of rows and columns of 2-D signal successively, which is show as equation (8) and equation (9) respectively.

$$F(u, v) = \sqrt{\frac{2}{R}} c(u) \sum_{r=0}^{R-1} \left\{ \sqrt{\frac{2}{C}} c(v) \sum_{c=0}^{C-1} f(r, c) \cos \left[\frac{\pi(2c+1)v}{2C} \right] \right\} \quad (8)$$

$$\cos \left[\frac{\pi(2r+1)u}{2R} \right]$$

$$f(r, c) = \sqrt{\frac{2}{R}} c(u) \sum_{u=0}^{R-1} \sum_{v=0}^{C-1} \left\{ \sqrt{\frac{2}{C}} c(v) F(u, v) \cos \left[\frac{\pi(2c+1)v}{2C} \right] \right\} \quad (9)$$

$$\cos \left[\frac{\pi(2r+1)u}{2R} \right]$$

The equation (8) and equation (9) is called conventional serial algorithm (CSA), and the separable transform form becomes the 2-D DCT core thought of serial computing.

From the equation (8) and equation (9), we can see that the operation of 2-D DCT needs $2N^3$ times multiplication and $2N^2(N-1)$ times addition if the DCT is applied directly without improvement. If the 2-D signal size is 8×8 , the total calculating amount is 1024 times multiplication and 896 times addition.

Set

$$T_{R \times R} = \begin{bmatrix} Q_{1,1 \times R} \\ Q_{2,1 \times R} \\ \vdots \\ Q_{R,1 \times R} \end{bmatrix} \quad P_{C \times C} = \begin{bmatrix} Q_{1,1 \times C} \\ Q_{2,1 \times C} \\ \vdots \\ Q_{C,1 \times C} \end{bmatrix} \quad (10)$$

Where,

$$Q_{(u+1),1 \times N} = [Q(0,u) \quad Q(1,u) \quad \cdots \quad Q(N-1,u)]$$

$$Q(r,u) = \sqrt{\frac{2}{N}} c(u) \cos \left[\frac{\pi(2r+1)u}{2N} \right]$$

The $Q(r,u)$ is called DCT transform kernel, the $Q_{(u+1),1 \times N}$ is called basis signal, and matrix T and matrix P are called row transform matrix and column transform matrix respectively. If the R and C both are equal to N , the matrix T and matrix P are the same and both called transform matrix.

Based on the transform matrix, the 2-D DCT can be described as follows.

$$F_{R \times C} = T_{R \times R} f_{R \times C} P_{C \times C}^T \quad (11)$$

$$f_{R \times C} = T_{R \times R}^T F_{R \times C} P_{C \times C} \quad (12)$$

The operation form as equation (9) and (10) calculate 2-D DCT point-by-point, and called serial computing too.

IV. 2-D SDCT ALGORITHM

The main idea of 2-D SDCT doesn't divide 2-D data into rows and columns, but to operate the 2-D signal as a whole to increase the computing speed.

A. The definition of transform basic matrix

In order to explore 2-D SDCT, a new transform matrix, T , is designed as equation (13).

$$T_{R^2 \times C^2} = \begin{bmatrix} Q_{11,R \times C} & Q_{12,R \times C} & \cdots & Q_{1C,R \times C} \\ Q_{21,R \times C} & Q_{22,R \times C} & \cdots & Q_{2C,R \times C} \\ \vdots & \vdots & \ddots & \vdots \\ Q_{R1,R \times C} & Q_{R2,R \times C} & \cdots & Q_{RC,R \times C} \end{bmatrix} \quad (13)$$

Where,

$$Q_{(r+1)(c+1),R \times C} = \begin{bmatrix} Q(0,0,r,c) & Q(0,1,r,c) & \cdots & Q(0,C-1,r,c) \\ Q(1,0,r,c) & Q(1,1,r,c) & \cdots & Q(1,C-1,r,c) \\ \vdots & \vdots & \ddots & \vdots \\ Q(R-1,0,r,c) & Q(R-1,1,r,c) & \cdots & Q(R-1,C-1,r,c) \end{bmatrix} \quad (14)$$

$$Q(u,v,r,c) = \sqrt{\frac{2}{R}} c(u) \cos \left[\frac{\pi(2r+1)u}{2R} \right] \sqrt{\frac{2}{C}} c(v) \cos \left[\frac{\pi(2c+1)v}{2C} \right]$$

The $Q(u,v,r,c)$ is called 2-D SDCT transform kernel, the $Q_{(r+1)(c+1),R \times C}$ is called basis signal, and the matrix T is called transform basic matrix. If the size of 2-D signal is 8×8 , the transform basic matrix is shown as fig.1.

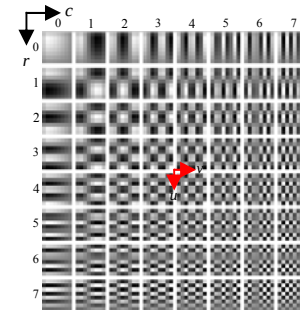


Fig.1 Transform basic matrix of 8×8 signal

Note: The range of transform basic matrix is adjusted to $[0,255]$ in order to improve visual effect.

B. 2-D SDCT Expression

According to the definition of “sub-matrix product” operation and “matrix superimposition product” operation, equation (6) and equation (7) can be rewritten as equation (15) and equation (16).

$$[F(u,v)]_{kl} = \sum_{r=0}^{R-1} \sum_{c=0}^{C-1} f(r,c) Q(u,v,r,c) = \sum_{r=0}^{R-1} \sum_{c=0}^{C-1} F(u,v) Q_{(u+1)(v+1)(r+1)(c+1)} \quad (15)$$

$$[f(r,c)]_{kl} = \sum_{u=0}^{R-1} \sum_{v=0}^{C-1} F(u,v) Q(u,v,r,c) = F \triangleright Q_{(r+1)(c+1),R \times C} \quad (16)$$

According to transform basic matrix, the definition of “sub-matrix product” operation and “matrix superimposition product” operation, equation (15) and equation (16) can be abbreviated further as equation (17) and equation (18).

$$F_{R \times C} = f_{R \times C} \triangleleft T_{R^2 \times C^2} \quad (17)$$

$$f_{R \times C} = F_{R \times C} \triangleright T_{R^2 \times C^2} \quad (18)$$

The equation (17) and equation (18) are SDCT expression form of 2-D DCT. Through analysis of equation (17), we can see that superimposing all sub-matrices can accomplish 2-D DCT after the product operation of each element of signal f and the corresponding sub-matrix of transform basic matrix T is completed. In other words, the transform result F can be expressed as weighted sum of the 2-D signal f and basis signal.

From the equation (18), we can see that the “sub-matrix product” operation of 2-D transform result F and a certain

sub-matrix (basis signal) can obtain a inverse transform result corresponding to the sub-matrix position. If the 2-D transform result F and each sub-matrix of transform basic matrix are performed one time “sub-matrix product” operation respectively, $R \times C$ results on corresponding position are obtained, that is, the inverse transform result, f . So, the 2-D DCT computation method using transform basic matrix is simple, and easy to understand.

For example, let the 2-D signal size be 8×8 , the transform process of DCT can be shown as fig.2 well [13, 14].

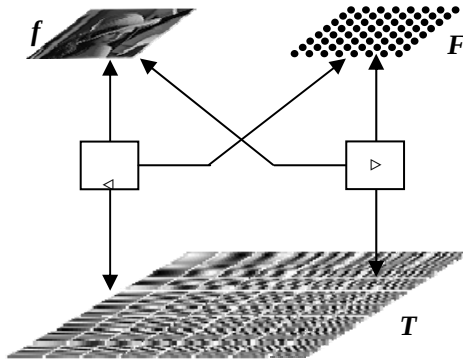


Fig.2 The operation sketch map of 2-D SDCT

In addition, the equation (17) and equation (18) can be rewritten as follows according to the definition of “sub-matrix element position” operation.

$$F_{R \times C} = f_{R \times C} \times T_{R^2 \times C^2}^L \quad (19)$$

$$f_{R \times C} = F_{R \times C} \triangleleft T_{R^2 \times C^2}^L \quad (20)$$

Similarly, the operation sketch map according to the equation (19) and equation (20) can be shown as Fig.3.

V. PARALLEL COMPUTING OF 2-D SDCT

The computing process of 8×8 signal as an example, the parallel computing of 2-D DCT is explained as follows.

A. Parallel computing of 2-D DCT

After the product operation between each element of f and the corresponding sub-matrix of T , superimposing all sub-matrices can accomplish 2-D DCT. In the process of operation, the product operation between an arbitrary element of f and sub-matrices of T can be processed simultaneously.

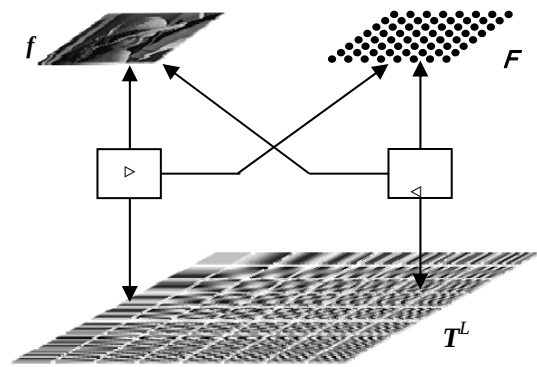


Fig.3 The operation sketch map of 2-D SDCT

The product operation of $f(0,0)$ and Q_{11} as an example, the product operation has no contact with other sub-matrix shown as Fig.4(a). We call this operation mode for internal parallel processing of 2-D DCT, and denote simply by Fig.4 (b).

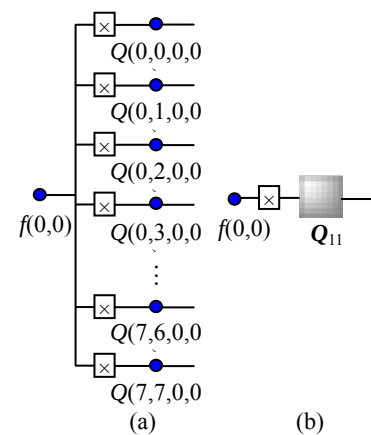


Fig.4 The sketch of 2-D IDCT internal parallel processing

In addition, the product operation of 8×8 elements of f and 64 elements of each sub-matrix can use parallel computing shown as Fig.5, which will generate 8×8 sub-matrices multiplied by corresponding elements of f . So all parallel multiply operations are processed in the same time, and the actual time-consuming of the multiplication is equivalent to 1 times multiplication time.

The 8×8 sub-matrices multiplied by corresponding elements of f only are added up to get the final transform result F . In order to solve this superimposing problem, grading parallel accumulation strategy is adopted shown as Fig.5. In Fig.5, every two product sub-matrices are added up to 32 addends at step 1, every two addends are added up to 16 addends at step 2. And so on, 8 addends at step 3, 4 addends at step 4, 2 addends at step 5 and 1 addends at step 6. So transform result F can be obtained by 6 levels of accumulation, and the actual time-consuming of the addition is equivalent to 6 times addition time. We call this operation mode for external parallel processing of 2-D DCT.

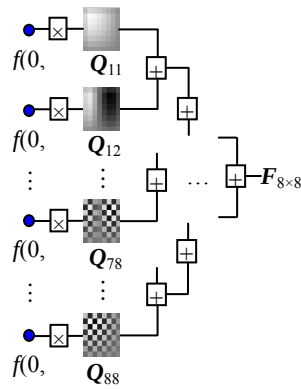


Fig.5 The sketch of 2-D IDCT external parallel processing

Generally speaking, because of the internal parallel processing and the external parallel processing, 2-D DCT calculation needs 1 times parallel multiplication and 6 level parallel additions. So the actual time-consuming is equivalent to 1 times multiplication and 6 times addition time.

B. Parallel Computing of 2-D IDCT

The principle and calculation process of 8×8 signal as an example, the parallel computing of 2-D IDCT is explained as follows. From Fig.3, we can see obviously that the transform matrix has 8×8 sub-matrixes, and the “sub-matrix product” operation between F and every sub-matrix generate a pixel value in corresponding position. So, the “sub-matrix product” operation between 2-D signal and 8×8 sub-matrixes generate 8×8 pixel value, that is, finish the 2-D signal IDCT computing.

In the process of operation, the “sub-matrix product” operation between F and 8×8 sub-matrixes can be processed simultaneously, and 8×8 sub-matrixes has noninterference as shown in Fig.6 below.

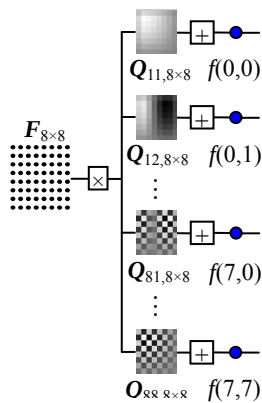


Fig.6 The sketch of 2-D DCT external parallel processing

The “sub-matrix product” operation between F and sub-matrixes $Q_{11,8 \times 8}$ as an example, the pixel result $f(0,0)$ can be obtained only based on the F and the sub-matrixes $Q_{11,8 \times 8}$ and has no contact with other sub-matrix. We call this operation mode for external parallel processing of 2-D DCT.

In addition, the product operation between 64 elements of the F and 64 elements of each sub-matrix can use parallel

computing also. The product operation between the F and sub-matrixes $Q_{11,8 \times 8}$ as an example, the calculating process is shown as Fig.7, which will generate 64 products.

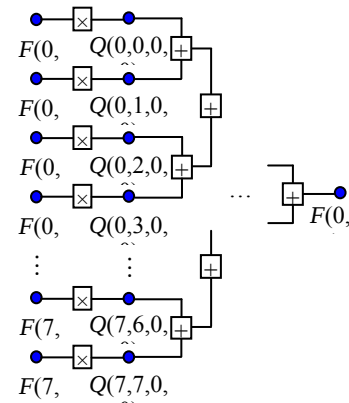


Fig.7 The sketch of 2-D DCT internal parallel processing

The 64 products only are added up to get the final result $f(0,0)$. In order to solve this problem, grading parallel accumulation strategy is adopted shown as Fig.7. In Fig.7, every two products are added up to 32 addends at step 1, every two addends is added up to 16 addends at step 2. And so on, 8 addends at step 3, 4 addends at step 4, 2 addends at step 5 and 1 addends at step 6. So result $f(0,0)$ can be obtained by 6 levels of accumulation. Other 63 transform results can be obtained by similar parallel computing described as the $f(0,0)$. We call this operation mode for internal parallel processing of 2-D IDCT.

Overall, because of the internal parallel processing and the external parallel processing, 2-D IDCT calculating needs 1 times parallel multiplication and six-step parallel additions. So the actual time-consuming equivalent to 1 times multiplication and six times addition time.

VI. ALGORITHM PERFORMANCE ANALYSIS

In order to explain the algorithm capability, a contrastive analysis is carried out between Sun algorithm^[3], CSA, Vetterli algorithm^[8], Feig algorithm^[9] and the 2-D parallel computing (2DPC) proposed in the paper. The equivalent calculating amount (ECA) of the four algorithms is listed as Table I.

Table I The Equivalent Calculation Comparison of 8×8 DCT

means	CAS		Vetterli		Feig		Sun		2DPC	
	×	+	×	+	×	+	×	+	×	+
ECA	1024	896	208	524	60	462	1	6	1	6

From table I, we can come to the conclusion safely that the equivalent multiplication and addition needed to accomplish 2-D DCT by the proposed algorithm reduced significantly. The equivalent multiplication needed by the proposed algorithm only is 0.098%, 0.48%, 1.67% of CAS, Vetterli algorithm, Feig algorithm respectively, and the equivalent addition only is 0.67%, 0.115%, 1.30%. But the calculating amount of the proposed algorithm and the Sun algorithm are the same. These outcomes can prove fully the

superior performance of the proposed algorithm.

From the operation process of 8×8 signal, we can see that the DCT computation amounts of $N \times N$ signal are equivalent to 1 times multiplication and a few additions, and the time required to complete addition is decided by the superposition steps. For example, the superposition steps needed equal to 2, 4, 4, 5, 6, 6 and 6 respectively if the size of 2-D signal is 2×2 , 3×3 , 4×4 , 5×5 , 6×6 , 7×7 and 8×8 . And so on, to $N \times N$ signal, the superposition steps needed is the power exponent of the minimum among such numbers which are all more than $N \times N$ and are power of 2.

In order to calculate and express conveniently, a operators, Δ , is defined. The operational criterions are described as follows.

$$a \Delta b = c \quad (21)$$

Where, c is the power exponent of the minimum among such numbers which are all more than a and are a power of b .

As an example of the calculation process of $7 \Delta 2$, the operational criterion of equation (21) is described in details. First, search such numbers which all are more than 7 and are a power of 2. Obviously, these numbers is 2^3 , 2^4 , 2^5 etc. Second, search the minimum among the numbers, that is, 2^3 . Finally, calculate the power exponent, that is, 3. And so on, we can see that $8 \Delta 2$ is equal to 3; $9 \Delta 3$ is equal to 2 etc. So $N \times N$ signal needs $(N \times N) \Delta 2$ steps parallel additions and the actual time-consuming is equivalent to $(N \times N) \Delta 2$ times addition time.

In addition, the multiplication times are not related to the size of 2-D signal, and only equivalent to 1 times multiplication. But the addition times are related to the size very closely. According to the calculation method indicated as equation (21), 2-D DCT parallel computing will reach optimum efficiency when N is integer power of 2.

VII. CONCLUSION

Aiming at the drawbacks of 2-D DCT serial computing, the traditional habitual thinking is broken, and a new algorithm, 2-D SDCT, is proposed in the paper. According to the operation rule of 2-D DCT, a new transform matrix and some new matrix operation methods are defined in the interest of convenient parallel computing. The 2-D DCT can be accomplished by parallel computing tidily based on the transform basic matrix. The 2-D DCT calculated by the proposed algorithm, 2-D SDCT, only needs less calculation time; the operation efficiency relative to serial operation is improved notably. In addition, 2-D

DCT parallel computing will reach optimum efficiency if the size of 2-D signal is an integer power of 2.

ACKNOWLEDGMENT

This work was supported by Projects of International Cooperation and Exchanges NSFC (Grant No. 609111301281), National Natural Science Foundation of China (Grant No. 60832002) and Scientific and technological development plan project of Jilin Province (Grant No. 20090302).

REFERENCES

- [1] Wenbang Sun, Hexin Chen, Hong Chen and Wenbing Sun. Research on 2-D SDCT Performance Based on Transform Basic Matrix. *Applied Mechanics and Materials*. 2011.63, pp523~527.
- [2] Chen W. A., Harrison C., Fralick S. C. A Fast Computational Algorithm for the Discrete Cosine Transform. *IEEE Transactions on Communications*, 1977, 25(9), pp1004~1011.
- [3] Sun Wenbang, Chen Hexin, Sun Wenbing. Research on Fast 2-D DCT Algorithm Based on Parallel Computing. *NCIS2011*, pp401~404.
- [4] Lee B G. A New Algorithm to Compute the Discrete Cosine Transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1984, ASSP-32(6), pp1243~1245.
- [5] Yin R X, Siu W C. A new fast algorithm for computing prime-length DCT through cyclic convolutions. *Signal Processing*, 2001, 81 (5), pp895~906.
- [6] Loeffler C, Ligtenberg A, Moschytz G. Practical Fast 1-D DCT Algorithms with 11 Multiplications. *Proc. of Intl. Conf. on Acoustics, Speech, and Signal Processing, ICASSP'89*, 1989, pp988~991.
- [7] Suehiro N. Fast algorithms for the DFT and other sinusoidal transform. *IEEE Transactions on Acoustics Speech and Signal Processing*, 1986, 34(3), pp642~644.
- [8] Vetterli M. Simple FFT and DCT algorithms with reduced number of operations. *IEEE Signal Processing*, 1984, 6(4), pp267~278.
- [9] Feig E., Winograd S. Fast algorithm for the discrete cosine transform. *IEEE Trans SP*, 1992, 40(9), pp2174~2193.
- [10] Tran T D. The BinDCT: Fast Multiplierless Approximation of the DCT. *IEEE Signal Processing Letters*, 2000, 7 (6), pp141~144.
- [11] Sun Wenbang, Chen Hexin, Sun Wenbing, Zhou Maiyu. Research on 2-D DCT Algorithm and Performance Based on Transform Basic Matrix. *Applied Mechanics and Materials Vols. 58-60 (2011)* pp 2570~2575.
- [12] Sun Wenbang, Chen Hexin, Chen Hong, Wang Zhiqiang. Research on 2-D SDCT Method Based on Parallel Computing. *CECNet2011*, 590~593.
- [13] Wenbang Sun1, 2, a, Hexin Chen1, b, Hong Chen2, c and Wenbing Sun. Research on 2-D SDCT Performance Based on Transform Basic Matrix. *Applied Mechanics and Materials Vols. 63-64 (2011)* pp 523~527.
- [14] Wenbang Sun1, 2, a, Hexin Chen1, b, Wenbing Sun3, c and Haiyan Tang. Research on SDCT Operation Based on Transform Basic Matrix. *Applied Mechanics and Materials Vols. 63-64 (2011)* pp 987~990.